Expandable Grids for Visualizing and Authoring Computer Security Policies

Robert W. Reeder[*] reeder@cs.cmu.edu
Lujo Bauer[*] lbauer@cmu.edu
Lorrie Faith Cranor[*] lorrie@cmu.edu
Michael K. Reiter[*†] reiter@cs.unc.edu
Kelli Bacon[‡] kbacon@gonzaga.edu
Keisha How[*] khow@cmu.edu
Heather Strong[*] hstrong@andrew.cmu.edu

[*]Carnegie Mellon University Pittsburgh, PA, USA
[†]University of North Carolina Chapel Hill, NC, USA
[‡]Gonzaga University Spokane, WA, USA

ABSTRACT

We introduce the Expandable Grid, a novel interaction technique for creating, editing, and viewing many types of security policies. Security policies, such as file permissions policies, have traditionally been displayed and edited in user interfaces based on a list of rules, each of which can only be viewed or edited in isolation. These list-of-rules interfaces cause problems for users when multiple rules interact, because the interfaces have no means of conveying the interactions amongst rules to users. Instead, users are left to figure out these rule interactions themselves. An Expandable Grid is an interactive matrix visualization designed to address the problems that list-of-rules interfaces have in conveying policies to users. This paper describes the Expandable Grid concept, shows a system using an Expandable Grid for setting file permissions in the Microsoft Windows XP operating system, and gives results of a user study involving 36 participants in which the Expandable Grid approach vastly outperformed the native Windows XP file-permissions interface on a broad range of policy-authoring tasks.

Author Keywords

security, policy, Expandable Grid, file permissions, visualization

ACM Classification Keywords

D.4.6 Security and protection, H.1.2 User/Machine systems, H.5.2 User Interfaces

INTRODUCTION

Access-control policies are fundamental to providing secure access to shared data. Good computer security is rooted in accurate policies that capture the intentions of their authors; inaccurate policies can lead to denial of service on the one hand and to compromised resources on the other. Humans determine these policies, and convey them, through some user interface, to the access-control systems that implement the policies. It is important that such user interfaces allow these humans, whom we call *policy authors,* to convey their policies accurately. However, research and everyday experience have shown that the user interfaces we have today for policy authoring are not usable.

As computing becomes increasingly collaborative, distributed, and pervasive, data becomes more available, and authoring policies to control access to this data becomes both more necessary and more challenging. In this environment, policy authoring increasingly falls to end users or non-technical people. While dedicated system administrators might be able to invest the time to learn and use complex user interfaces, this rising class of novice and occasional policy authors cannot be expected to do the same.

Despite the need for usable policy-authoring interfaces, there is evidence that widely-used policy-authoring interfaces are prone to serious errors. The "Memogate" scandal, in which staffers from one political party on the United States Senate Judiciary Committee stole the opposing party's confidential memos from a file server that the two parties shared, was caused in part by an inexperienced system administrator's error using the Windows NT interface for setting file permissions [9]. The administrator had just finished college and had not completed advanced training in system administration. Good and Krekelberg showed that the Kazaa file-sharing user interface misled users into unintentionally sharing confidential data [2]. A study on the usability of file permissions interfaces showed cases in which users of the Windows XP file permissions interface made errors that exposed files to unauthorized access [5].

One of the main problems with today's policy-authoring user interfaces is that the dominant model they use for displaying policies—the *list-of-rules* model—is deficient. List-of-rules interfaces display a list of the rules that comprise a policy, but only allow authors to select a single rule at a time for viewing or editing. A rule is merely a portion of a policy, so when a rule is viewed in isolation, important contextual information about the remainder of the policy is unavailable to the policy author. For example, rules may conflict with one another, so users need information about other rules to understand how a given rule will be affected by others. As another example, one access rule may apply to a group of objects, and another rule may state what objects are in that

CHI 2008, April 5 - 10, 2008, Florence, Italy.

group; the group membership rule may be crucial to understanding the effects of the access rule on specific objects. List-of-rules interfaces do not give any indication of how rules interact with one another; instead it is left to the policy author to figure out which rules will interact and how. This makes policy authoring difficult and error-prone, especially for novice and occasional authors.

To address the need for better policy-authoring interfaces, we introduce a new model for displaying and editing policies: Expandable Grids. An Expandable Grid is a matrix-based visualization of a policy, in which *principals* (the users of a system in which data is shared) are displayed in an interactive tree along one axis of the matrix, *resources* (the data to which access is to be controlled) are displayed in an interactive tree along the other axis, and the access allowed to each principal for each resource is shown in colored boxes in the matrix at the intersection of the two trees. Expandable-Grid-based policy-authoring interfaces show the *effective access* (i.e., the combined effect of all applicable rules) each principal will have to each resource, so that policy authors need not combine rules in their heads to figure out who will be able to access what. Expandable Grids can be viewed as an adaptation of Lampson's access matrix [4] to enable automated policy management.

We implemented an interface based on the Expandable Grid concept for setting file permissions in the Windows NTFS file system. We ran a user study to compare authoring performance on a wide range of authoring tasks between our interface and the native Windows XP file-permissions interface. The Expandable Grid interface performed vastly better in our study; for example, the Expandable Grid interface achieved a 100% accuracy rate versus a 6% accuracy rate for the Windows interface for one task, and, on another task where accuracy rates were similar, achieved a 70% reduction in time-to-task-completion compared to Windows. We show specifically why the Expandable Grid interface performed better and conclude that it is likely a better option than the list-of-rules model for policy-authoring interface designs for file permissions and other security-related applications.

PROBLEM DESCRIPTION

In this section, we discuss file permissions as an example of a policy-authoring domain, list the fundamental operations a policy-authoring user interface should support, and show why the list-of-rules model leads to error-prone policy-authoring interfaces.

File permissions

In a basic file access-control system, there are resources, principals, and actions. Resources include files and folders, the data to which access is to be controlled. Principals are system users and groups of system users. Actions are operations that may be performed on resources, such as "read," "write," "execute," and "delete." A file permissions policy is a set of rules that state what principals may perform what actions on which resources. Rules may allow or deny access. For example, a rule may state that the user "jsmith" is allowed to read the file "budget.xls," or that the user "mary" is denied access to write to the file "privateData.txt." Since a file system may have many principals, file access control systems typically provide a means for constructing *groups*, which are collections of users. For instance, users might be placed into groups or roles according to their position in an organization, such as "Managers," and "Employees." Similarly, files may be grouped into hierarchical folders. Groups and folders allow policy authors to write rules that apply to all components of a composite value, such as a rule that denies delete capability to all employees for files in a folder called "Critical files."

Fundamental policy-authoring operations

There are four fundamental operations a policy-authoring interface must support to be successful:

1. *Viewing policy.* Policy-authoring interfaces should make it easy to look up what will happen when a given user tries to access a given resource. Furthermore, they should include the ability to check general properties the author would like the policy to have, such as "no one should be allowed to delete this file," or "no remote users should have access to this folder."
2. *Changing policy.* Policy-authoring interfaces should allow for making rules, which alter the access a policy will allow. In addition, a policy should be editable, since policy-authoring is often done in response to some situation, such as access being denied to someone who needs it.
3. *Viewing composite value memberships.* Composite values (i.e., groups of users or directories of files) make it much easier to specify policy rules that apply to many individual input values, such as "All of my friends are allowed to read files in my 'Music' directory." When a policy author is setting a rule involving one or more composite values, it is important to know just what the composite values contain. A policy-authoring interface should show the individual members of composite values.
4. *Detecting and resolving conflicts.* Rule conflicts occur when two rules apply to a particular request for resources. This may happen, for example, if a user is explicitly denied access to a resource, but is a member of a group that is explicitly allowed to access that resource. While policy-based systems typically adopt some means of resolving such conflicts, some past work has shown that users have difficulty understanding how conflicts are resolved [5]. As a result, conflicts may lead to policies that do not implement the author's intentions. A policy-authoring interface should make sure that conflicts do not cause policies to deviate from authors' intentions.

List-of-rules model

File permissions interfaces based on the list-of-rules model are common. The most widely distributed, and the one we study in this paper, is the Windows XP file permissions interface, but the model is also used in Linux open source file permissions managers and in the file permissions interface for Mac OS X Server. The list-of-rules model is also used in policy-authoring tools for other domains such as firewall policies and enterprise privacy policies.

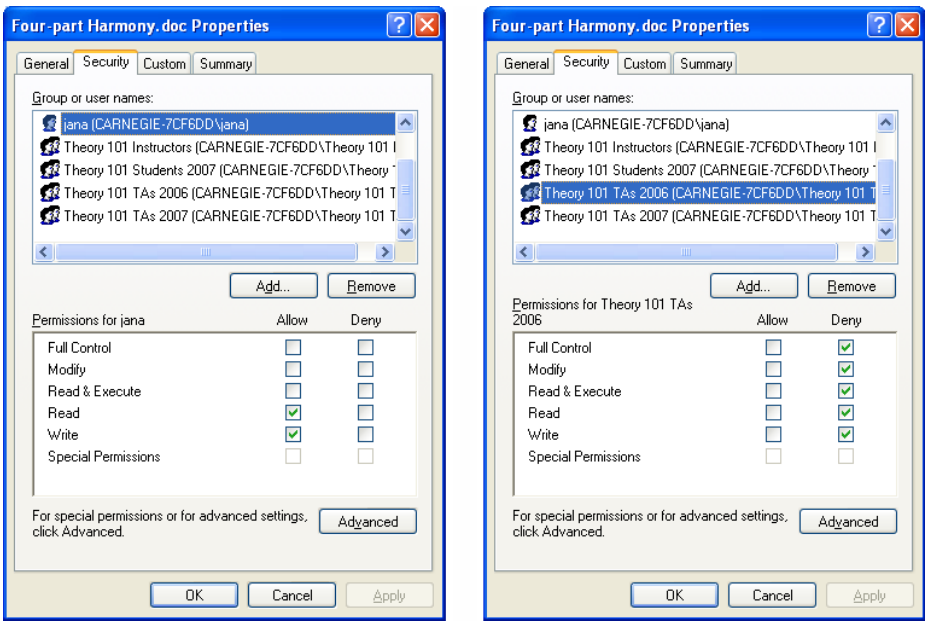

Figure 1. The Windows XP file permissions interface, an example of a list-of-rules policy-authoring interface. These screenshots give an example of why it is difficult for users to understand an effective policy in the presence of a rule conflict. See text for further explanation.

The Windows XP file permissions interface provides a good example of a list-of-rules policy-authoring interface and its limitations. To describe the limitations of the list-of-rules approach, it is convenient to start with an example task. Here, we describe the Jana task, one of the 20 tasks we designed for our user study, which is described below. Our statement of the Jana task reads as follows:

> Jana, a Theory 101 TA, complained that when she tried to change the Four-part Harmony handout to update the assignment, she was denied access.
>
> Set permissions so that Jana can read and write the *Four-part Harmony.doc* file in the *Theory 101\Handouts* folder.

Figure 1 shows the state of the Windows XP file permissions interface after a common error that policy authors make. Authors create a rule stating that Jana is allowed read and write access to the *Four-part Harmony.doc* file. Looking at the interface as pictured in the left-hand screenshot in Figure 1, it appears Jana is now allowed to read and write the file. What policy authors miss, however, is that Jana is a member of the group Theory 101 TAs 2006; there is a rule, as can be seen in the right-hand screenshot in Figure 1 that denies this group access to read and write the file. Thus, two rules apply to Jana, one allowing her access and one denying her access; this situation is a rule conflict. In Windows, rules that deny access take precedence in conflicts with rules that allow access, so in Jana's case, she is denied access to the file. To understand Jana's situation, the policy author must understand three things: first, that Jana is in the group Theory 101 TAs 2006; second, that Theory 101 TAs 2006 are denied write access to the file; and third, that Deny permissions take precedence over allow permissions. Once these three items are understood, they need to be remembered and applied to determine the correct method for completing this task: removing the deny rule for Theory 101 TAs 2006 so that the rule allowing Jana access will become operative, while the other members of the group will still be denied access to the file by default. However, the Windows list-of-rules interface makes it difficult for the policy author to obtain these three pieces of information. Group membership information is only available in the Computer Management application, which is wholly separate from the file permissions interface, and thus rarely found; the rule applying to Theory 101 TAs 2006 is easily overlooked; and the deny-takes-precedence rule essentially has to be inferred. It is difficult for novice or occasional policy authors to complete this task correctly, even though it is a fairly typical scenario.

EXPANDABLE GRIDS

To address the limitations of the list-of-rules model, we introduce Expandable Grids. An Expandable Grid is a conceptual visualization for displaying security policies to policy authors and end-users in a graphical format. Expandable Grids show precisely what a policy allows or does not allow in a matrix with hierarchical axes that can be expanded or contracted to show more or less policy detail. Expandable Grids do not require policy authors to determine subtle interactions among rules; they show the holistic effect of all policy rules, i.e., the effective policy for each principal and resource, thus relieving the burden on policy authors of figuring out how rules interact with each other. Expandable Grids also have the advantage of showing an overview of the entire policy, thus making it easy, for certain tasks, to see what portions of the policy need to be examined in detail.

We have designed and implemented a file permissions policy-authoring user interface based on the Expandable Grid idea. A screenshot of our interface can be seen in Figure 2. The

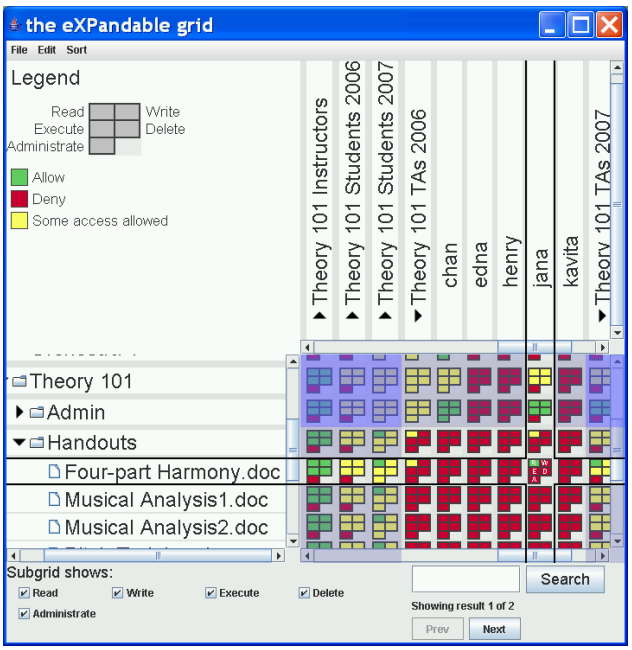

Figure 2. Screenshot of our Expandable Grid interface when the Jana task has been half-completed.

tree along the vertical axis at the left of the interface shows the resources in a file system. The rotated tree along the horizontal axis at the top of the interface shows the principals. At the intersection of these two trees is a grid that shows the access each principal has to each resource. Grid cells each correspond to one principal and one resource. Each grid cell is further subdivided into a "subgrid," a large square divided into smaller boxes. The subgrids allow the interface to show a third policy dimension, as long as that dimension has only a handful of values. In our interface, the subgrid represents five different types of access. The upper left box in each group of five boxes indicates read access, the upper right box indicates write access, the middle left box indicates execute access, the middle right box indicates delete access, and the lower box indicates administrate access. Green boxes (which appear as a medium grey in greyscale) indicate access that is allowed, red boxes (which appear dark grey in greyscale) indicate access that is denied, and yellow boxes (which appear light grey in greyscale) indicate that items lower in one or both trees have a mixture of allowed and denied access.

The screenshot in Figure 2 shows an intermediate state the interface would be in when the Jana task has been half-completed. The grid cell at the intersection of "jana" and the "Four-part Harmony.doc" file is highlighted in the crosshairs. In the highlighted grid cell, the upper-left box is green, indicating that Jana has already been given read access to the "Four-part Harmony.doc" file, while the upper-right box is red, indicating that Jana does not yet have write access. Clicking on the upper-right box will give Jana write access and cause the box to turn green. In contrast to the checkboxes in the Windows file permissions interface, the colored boxes of the grid indicate the actual access a user will have to a resource, after all policy settings have been taken into account. When a red box is clicked on, the policy is changed so that Jana is now allowed access, and the box turns green; there is no need to be aware of other rules that may interfere with the intention to allow her access. Thus, it is much easier to complete the Jana task using the Expandable Grid interface and to be certain that it has been completed correctly.

When policy is set at the group level, its effects are immediately propagated to the members of the group, so that the access indicated in the grid is always the access that will be given. For example, again referencing Figure 2, if the upper-right box in the grid cell at the intersection of the group "Theory 101 TAs 2006" and the "Four-part Harmony.doc" file were clicked to change it from red to green, the upper-right boxes for all the group's users—"chan," "edna," "henry," "jana," and "kavita"—would all turn green.

New policy semantics

In order to ensure that clicking on the colored boxes in the Expandable Grid interface consistently has the same effect, we changed some of the semantics of Windows file permissions policies. Specifically, we changed the means by which rule conflicts are resolved. In Windows, when Jana is part of one group that is allowed access to a file and another that is denied access to that file, the deny rule takes precedence. If we were to retain the Windows semantics, clicking on a red box where a deny rule applies would leave the box red, rather than changing it to green as the policy author would expect. Even though clicking on the red box would add an allow rule, the box would stay red because the existing deny rule would still take precedence over the new allow rule. By our revised semantics, the most recently created rule takes precedence. Thus, if the box corresponding to Jana's read access for the "Four-part Harmony.doc" file is red because of her membership in Theory 101 TAs 2006, but a policy author clicks on the box to make a rule that allows Jana access, this new rule will take precedence over the deny rule from the group and the box will turn green; Jana will now be allowed to access the file.

Our revised semantics makes for much easier resolution of rule conflicts than is possible in Windows. In Windows, resolving the Jana task requires either removing the deny permission not only for Jana, but also for all the other users in the Theory 101 TAs 2006 group, or removing Jana from the group entirely. Neither solution may be desirable, and in any case, policy authors find this dilemma confusing, if they even understand it at all [5].

Our semantics does have a potential drawback, however. The Imani task we designed for our study, which is described below, illustrates this drawback. Our description of the Imani task reads as follows:

> ....Two weeks ago, a scandal erupted in the Music Department... someone had been selling exam answers to the students! You suspected Imani, your fellow TA. Like the responsible administrator you are, you set file permissions so that Imani could not access the Answers folder under any circumstances.
>
> Now, it's exam grading time, and your fellow TAs need access to the answers. Make sure they have access to

the Answers folder, but be sure not to let Imani have access.

Set permissions so that TAs 2007 (except for Imani) can read the Answers folder.

We set up the policy beforehand to have a deny rule applying to Imani for the Answers folder. In the Windows semantics, if a participant set a rule allowing the group TAs 2007 (which includes Imani) to read the Answers folder, Imani would still be denied access to the folder, even if the participant completely forgot to make an exception for Imani. In our semantics, if the participant set a rule allowing the group TAs 2007 to read the Answers folder, this rule would override the pre-existing rule denying Imani access, and, unless the participant went back and restored the rule denying Imani access, Imani would be allowed access.

Summary of Expandable Grid features

Our Expandable Grid interface has several features that we expect to provide advantages over the list-of-rules Windows XP file permissions interface. In particular, the Grid interface has these features:

- *Whole policy.* The Grid shows the whole policy, including principal/resource combinations for which there is no explicit rule.
- *Effective policy.* The Grid shows the effective policy, while Windows merely shows component rules.
- *Group membership information.* The Grid integrates group membership information into the file permissions display, while Windows puts it in a separate application from the file permissions interface.
- *Simple changes.* The Grid requires a simple click on a colored box to change a permission, while the Windows interface requires adding a new rule to its list.
- *New policy semantics.* The Grid's new policy semantics allows for easy conflict resolution by simply clicking on a colored box, the same way any other policy change would be made.
- *Visual pop-out.* The Grid allows for easy detection of anomalous permissions that visually pop out from the rest of the policy display.

Additional functionality

Our implementation of the Expandable Grid includes two additional functions worth mentioning: highlighting and search. Highlighting can be seen in Figure 2. The crosshairs follow the mouse pointer when it is positioned over the grid to help authors line up a grid cell with the principal and resource for which they wish to view or edit a rule. The search function can help users navigate a potentially large policy. Names of principals or resources can be typed into the search box. If the search term matches all or part of the name of a principal or resource, the principal or resource will be highlighted. If there are multiple search hits, the first will be highlighted, and other hits can be stepped through using the "Prev" and "Next" buttons beneath the search box.

RELATED WORK

Several authors have addressed usability concerns in their own systems for authoring access-control policies. Two systems, SPARCLE and the HP Select Access Policy Builder, discussed below, have implemented matrix-based policy visualizations, but have not published any evaluation of the matrix-based visualization approach as compared to the list-of-rules approach.

Lampson was the first to describe access control matrices as a framework for teaching and reasoning about access control [4]. However, Lampson was not concerned with user interfaces, and so did not discuss how access control matrices could be used interactively to author policies.

Zurko et al. designed the Adage system, a policy-based access control system for distributed computing systems [8, 11]. Adage included the Visual Policy Builder, a graphical user interface for supporting policy authoring. Although Adage included extensive functionality to support all of the fundamental policy-authoring operations listed above, it still used a list-of-rules approach: users could view single rules or lists of rules, but not the full policy implied by the rules. In fact, Zurko's usability testing revealed the need for a policy overview, and she suggested that research in "dynamic visualization" could be applied to policy visualization [10].

The HP Select Access Policy Builder is an application for authoring enterprise security and privacy policies [6]. It includes a user interface that provides a visualization of a policy in a matrix indexed by principals on one axis and resources on the other. This is very similar to the idea of Expandable Grids. However, the HP Policy Builder apparently has no provision for showing additional input dimensions, such as actions (which we show using the subgrid). Moreover, no user study evaluating the HP Policy Builder has been published.

Karat et al.'s SPARCLE system is concerned with the problem of enterprise privacy policy authoring [3]. SPARCLE's most notable feature is its natural-language input mechanism for authoring policy rules, but it also provides a table-based policy visualization. SPARCLE's table visualization does not show hierarchical structure along the axes and its scalability is limited because it uses space-consuming text rather than graphics in the table cells.

One of the precursors of the current work was Maxion and Reeder's Salmon system [5] for authoring file permissions in Windows XP. The key insight behind Salmon was to show the effects of rule interactions to policy authors. However, Salmon remained a list-of-rules interface; the present work builds on the Salmon insight to show effective policy in a large visualization.

Cao and Iverson presented Intentional Access Management (IAM) systems for access control [1]. They define IAM systems as systems that automatically convert users' intentions into policy rules. They demonstrate that their IAM approach can reduce policy authoring errors compared to an unenhanced list-of-rules policy editor, but they do not provide any means for visualizing a full policy.

Rode et al. designed Impromptu, a file sharing application that includes a visualization-based user interface for specifying file sharing policies [7]. The Impromptu visualization depicts users, files, and actions on a pie chart. The visualization is limited, however, to showing policies with small groups of users (on the order of 6), and apparently cannot scale to show policies with many users.

USER STUDY METHODOLOGY

We ran a laboratory user study with 36 participants to compare our Expandable Grid interface against the native Windows XP file permissions interface. We used a between-participants design, so 18 participants used the Grid and 18 used Windows; participants were randomly assigned to an interface condition. Each participant completed 20 tasks related to viewing or changing file permissions for a hypothetical Windows NTFS file server. The tasks are described in the "Task design" section below. We measured accuracy and time-to-task completion for each task.

Participants

We recruited 36 undergraduates in technical majors (science, engineering, and mathematics) to participate in the study. Ten were female. Participants were all daily computer users, but had never served as system administrators. Thus, our participant pool was consistent with our target demographic of novice or occasional access-control-policy authors, and was of a similar demographic as the system administrator of the Memogate scandal. While we believe it is important to make access-control interfaces usable for non-technical users as well as technical users, we leave testing the Grid on non-technical users as future work.

Experimental setup

Participants worked on a laptop running the Windows XP Professional operating system. Task statements were presented in a Web browser and were available on screen throughout the task. For participants using Windows, we started the Computer Management interface in a state that allowed for viewing system users and groups and opened Windows Help files related to setting file permissions. For participants using the Expandable Grid, the Grid application window was started at an initial size of 800x740 pixels. The laptop screen resolution was 1280x768.

The data we collected included the policies people created and screen video and audio of participants thinking aloud.

Task design

We designed a broad range of tasks to test each of the fundamental policy-authoring operations in a variety of contexts and for both small- and large-scale policies. We created a scenario in which the participant was a teaching assistant (TA) in a music department, and was assigned to be the administrator of a file server that stored materials for classes in the department. We chose this scenario because it would be familiar to our participants while allowing for plausible tasks involving file permissions policies. The 20 tasks were designed around this scenario. Of these 20 tasks, 10 involved "small-scale" file permissions policies (involving roughly 50 principals and 50 resources) and 10 involved "large-scale" policies (roughly 500 principals and 500 resources). Each small-scale task had a large-scale parallel, so there were 10 pairs of tasks that varied in difficulty and in the fundamental operation they were testing. Also, each task pair tested one or more of the specific advantages or drawbacks we expected the Grid interface to have compared to the Windows interface. The following list describes what each of the ten task pairs required participants to do and which features (with reference to labels listed in the "Summary of Expandable Grid features" section above) of the Grid interface were tested by each:

1. *Training*: Make a simple policy change. The two tasks in this pair served to help the participant learn the interface they were using and get used to the scenario.
2. *View-simple*: Determine a user's access to a file when the user was inheriting access from a group. These tasks tested the Grid's *whole policy* and *group membership information* features.
3. *View-complex*: Determine a user's access to a file when a rule conflict is present. In this rule conflict, one rule applying to a specific user and one rule applying to a group of which that user is member were in conflict. These tasks tested the Grid's *effective policy* feature.
4. *Change-simple*: Change a principal's access to a resource. These tasks tested the Grid's *simple changes* feature.
5. *Change-complex*: Make multiple changes to a policy, including one involving a rule conflict. These tasks tested several of the Grid's features in one task, including the *whole policy*, *effective policy*, *group membership information*, *simple changes*, and *new policy semantics* features.
6. *Compare-groups*: Search for the intersection between two groups' members. These tasks tested the Grid's *group membership information* feature.
7. *Conflict-simple*: Detect and resolve a conflict in which the allow access a user is inheriting from a group should be overridden with an explicit deny rule. These tasks tested the Grid's *effective policy* and *simple changes* features.
8. *Conflict-complex*: Detect and resolve a conflict in which a user is allowed access through one group but denied access through another. The Jana task, discussed above, was the large-scale conflict-complex task. These tasks tested the Grid's *effective policy*, *simple changes*, and *new policy semantics* features.
9. *Memogate*: Detect that an unauthorized group has been given access to a resource. Our configuration for these tasks mimicked the vulnerability that made the Memogate scandal possible by including a policy rule that gave full access to all users for all files. These tasks tested the Grid's *visual pop-out* feature, because the unauthorized access causes the Grid to show a large area of green squares where red is expected, while Windows requires stepping through all rules to find the offending one.
10. *Precedence*: Allow a group to access a resource, but make an exception for one individual in the group. The purpose of these tasks was to demonstrate the drawback of our new policy semantics (the Imani task, discussed above, was the small-scale precedence task).

Table 1. Task statements given to participants for the small-scale conflict-complex and change-complex tasks, 2 of 20 tasks used in the user study.

Conflict complex
Nigel, another TA, has complained that when he was trying to edit the course calendar to change the due date for an assignment, he was denied access.
Set permissions so that *Nigel* can *read and write* the *calendar.scd* file.
Change complex
The gradebook file is highly sensitive. Instructors and TAs should be allowed to read it or write to it, but not delete it. And of course no students should be allowed to access the gradebook in any way.
Make sure *instructors* and *TAs 2007* can *read* and *write* the *gradebook.xls* file.
Make sure *instructors* and *TAs 2007* cannot *delete* the *gradebook.xls* file.
Make sure *Students 2007* cannot access the *gradebook.xls* file in any way.

Examples of the text of task statements presented to participants can be seen in Table 1.

Some tasks, such as the tasks that involved viewing policy or group memberships, were multiple choice questions, so scoring was straightforward. The other tasks required making policy changes. For some of these tasks, there were multiple approaches to completing the task; in all cases, we scored a participant's final policy for a task according to whether they gave the access specified in the task statement; any extraneous policy settings were ignored.

Procedure

Our experimenter read instructions for our teaching-assistant scenario to participants. After reading these instructions, our experimenter read brief training materials that explained how to search for users and files, how to view permissions, and how to change permissions using the interface to which a participant was assigned. The experimenter also walked the participant through one full task as part of the training. For Windows participants, the training also covered how to use the Computer Management application and how to use the Windows Help files, since we considered those part of the file permissions interface; no help files were available to Grid participants. Training took about 3.5 minutes for Grid participants and about 5.5 minutes for Windows participants.

Participants then began completing tasks. Before each task, the experimenter brought up the interface in a preconfigured state tailored to each task. Task statements were then presented to participants in a Web browser on screen. Participants were asked to think aloud while they worked on the tasks. Participants completed the 10 small-scale tasks first, then the 10 large-scale tasks. Each set of 10 tasks always started with the training task, but the order of the remaining 9 tasks were counterbalanced across participants using a Latin square design to guard against ordering effects.

RESULTS

Our results are in the form of accuracy rates and mean time-to-task-completion for each interface condition and for each task. Accuracy rates represent the proportion of participants in each condition who correctly completed the task. The mean time-to-task-completion is computed only over the task completion times of those participants who successfully completed each task. Accuracy results can be seen in Figure 3. Timing results can be seen in Figure 4. Note that results are only shown for 18 tasks, 9 at each scale, because we excluded the two training tasks from analysis.

Experiment-wide results

For each metric, accuracy and time-to-task-completion, we performed an experiment-wide test of the hypothesis that the Grid interface performed better than the Windows interface over all tasks. We used logistic regression to test for a significant difference in accuracy scores between the Grid (overall accuracy 83.6%) and Windows (overall accuracy 56.5%) interfaces across all tasks. We included interface, scale, and an interface*scale interaction as factors in our logistic regression model. The model gave an intercept of 1.52 and coefficients for interface, scale, and interface*scale of -1.22, 0.23, and -0.30. The corresponding odds ratios were 0.29, 1.25, and 0.74. A Wald test of the hypothesis that the interface coefficient was not equal to 0 was significant at the 0.05 level ($Z = 7.43, p < 0.001$), suggesting that there is an effect of interface on accuracy. The direction of the interface coefficient indicates that higher accuracy scores were achieved with the Grid interface. Wald tests of the hypothesis that the other coefficients were not equal to 0 were not significant ($Z = 0.75, p = 0.45; Z = -0.80, p = 0.42$), suggesting that the Grid interface's superior accuracy rates compared to the Windows interface are not affected by scale.

We used a general linear model with log-transformed time-to-task-completion as the response and interface, scale, and an interface*scale interaction as factors in the model and found significant effects of interface ($F(1, 450) = 91.15, p < 0.001$), scale ($F(1, 450) = 49.27, p < 0.001$), and interface*scale ($F(1, 450) = 20.81, p < 0.001$) on time-to-task-completion. The directions of the effects indicate that using the Grid (M=53.0 seconds, σ=37.5) led to faster task completion than using Windows (M=88.3 seconds, σ=62.7), that large-scale tasks (M=79.4 seconds, σ=51.8) took longer to complete than small-scale tasks (M=55.2 seconds, σ=49.7), and that the slowdown from small-scale tasks to large-scale tasks was greater for the Grid than for Windows.

Task-by-task results

We followed up the experiment-wide tests with a series of planned tests for significant differences in the accuracy rate and mean time-to-task-completion between the Grid interface and the Windows interface for selected tasks. We conducted these tests based on our hypotheses as to which interface would perform better in each task. Specifically, we hypothesized that we would see more accurate performance using the Grid for the view-simple, view-complex, change-complex, conflict-simple, conflict-complex, and Memogate tasks. We hypothesized that we would see more accurate performance using Windows for the precedence tasks, which were designed to test the drawback in the Grid's new policy semantics. We expected the change-simple and compare-groups tasks to be easy to complete with both interfaces, so we did not test for significant differences in accuracy for those tasks. For the time-to-task-completion data, we hypothesized that the Grid would perform better for all tasks except the precedence tasks, for which we conducted no test because we expected the Grid's drawback to affect accuracy, but to have little effect on time. In all, we had 30 hypotheses to test, but could only test 27 due to insufficient timing data for tasks that were correctly completed by too few Windows participants.

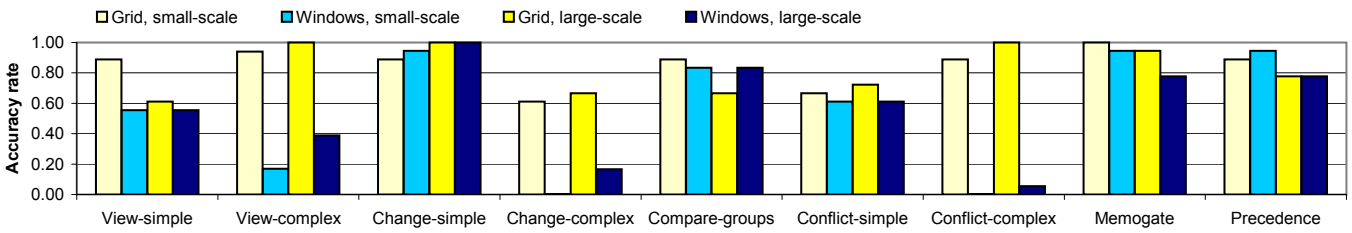

Figure 3. Accuracy results, showing proportion of participants correctly completing each task with Grid and Windows interfaces.

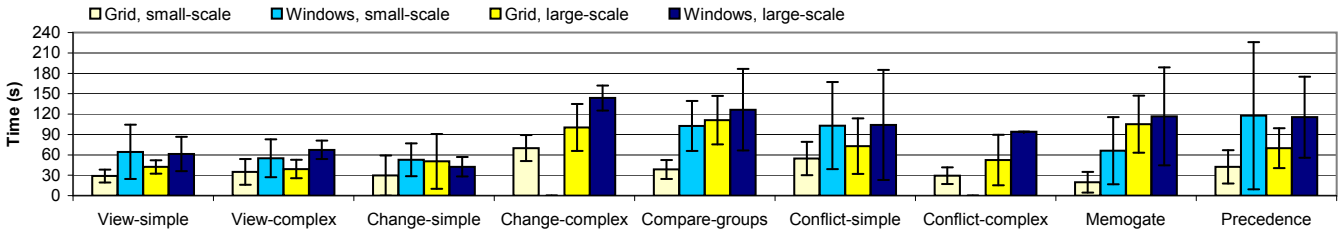

Figure 4. Time-to-task-completion results, showing mean time-to-task-completion, in seconds, for participants who successfully completed each task with Grid and Windows interfaces. Error bars show +/- one standard deviation.

To test our hypotheses, we conducted 14 one-sided Fisher's Exact Tests on the accuracy rates and 13 one-sided t-tests on the mean times-to-task-completion. We log-transformed the time data before testing to ensure that it was normally distributed, as assumed by the t-test. Since we applied 27 total statistical tests, we applied the Benjamini-Hochberg multiple testing correction to keep down the probability of a Type I error in our testing. The correction gave an adjusted per-test α of 0.024, so we considered only tests with p-values at or below this adjusted threshold to be significant.

Results of the tests are summarized in Tables 2 and 3. The 27 tests yielded 14 significant results, all in favor of the Grid. The 14 significant results included 6 significant differences in accuracy rates and 8 significant differences in mean time-to-task-completion. There was at least one significant result for every task pair except the precedence task pair.

Table 2. Summary of statistical tests for significant differences in accuracy rate for Grids (a_G) and accuracy rate for Windows (a_W) for each task. For all tests except precedence task tests, the hypothesis tested was $a_G > a_W$. For the precedence tests, the hypothesis tested was $a_G < a_W$. The p-values shown are from one-sided Fisher's Exact Tests; p-values below the $\alpha = 0.024$ rejection threshold are shaded and highlighted in bold, indicating significant tests.

Task pair	Small-scale			Large-scale		
	a_G	a_W	p-value	a_G	a_W	p-value
View-simple	0.89	0.56	$p = 0.03$	0.61	0.56	$p = 0.50$
View-complex	0.94	0.17	**$p < 0.001$**	1.00	0.39	**$p < 0.001$**
Change-simple	0.89	0.94	*No test*	1.00	1.00	*No test*
Change-complex	0.61	0.00	**$p < 0.001$**	0.67	0.17	**$p = 0.003$**
Compare-groups	0.89	0.83	*No test*	0.67	0.83	*No test*
Conflict-simple	0.67	0.61	$p = 0.5$	0.72	0.61	$p = 0.36$
Conflict-complex	0.89	0.00	**$p < 0.001$**	1.00	0.06	**$p < 0.001$**
Memogate	1.00	0.94	$p = 0.5$	0.94	0.78	$p = 0.17$
Precedence	0.89	0.94	$p = 0.5$	0.78	0.78	$p = 0.65$

DISCUSSION

Our results demonstrate that the Expandable Grid interface allows authors to complete tasks more accurately and faster than does the Windows file permissions interface. Because the tasks selected span the range of fundamental policy-authoring operations and cover policies of different sizes, we believe that these results will hold over a broad range of contexts, tasks, and policy sizes in real applications.

Our expectations about the advantages of the Grid's features were borne out by the statistical results of our user study. In particular, our study demonstrates higher accuracy for the Grid compared to Windows for the view-complex, change-complex, and conflict-complex tasks that tested the Grid's *whole policy*, *effective policy*, *group membership information*, *simple changes*, and *new policy semantics* features. The study also demonstrates faster times-to-task-completion for tasks that tested these features as well as the small-scale Memogate task, which tested the *visual pop-out* feature.

The precedence tasks were designed to test the drawback (explained above in the "New policy semantics" section) to the Grid's new policy semantics, but we did not see significantly poorer performance with the Grid in these tasks. In these tasks, participants had to allow access to a group but keep an exception that denies access to an individual. Our results suggest that the Grid's policy visualization may mitigate the drawback by helping policy authors realize when they are overriding an exception they wish to keep and allowing them to restore the exception easily.

Our results suggest that the Grid accuracy gains hold up for both small- and large-scale policies. However, our statistical analysis did reveal an interface*scale effect that shows Grid performance slowing down more than Windows performance as policies move from the small to the large scale. We see at least two possible explanations for this effect. First, it

Table 3. Summary of statistical tests for significant differences in mean time-to-task-completion, in seconds, between successful Grid (G) and Windows (W) participants for each task. For each test, the table shows means (M) and standard deviations (σ) for each interface, and t-statistics and p-values resulting from one-sided t-tests; p-values at or below the $\alpha = 0.024$ rejection threshold are shaded and highlighted in bold, indicating significant tests.

	Small-scale			Large-scale		
Task pair	Mean, standard deviation	t-statistic	p-value	Mean, standard deviation	t-statistic	p-value
View-simple	G:$M = 28.8, \sigma = 9.5$ W:$M = 64.6, \sigma = 39.9$	$t(14) = -4.37$	$\mathbf{p < 0.001}$	G:$M = 42.3, \sigma = 9.9$ W:$M = 61.3, \sigma = 25.3$	$t(14) = -2.17$	$\mathbf{p = 0.024}$
View-complex	G:$M = 34.8, \sigma = 19.0$ W:$M = 55.0, \sigma = 28.0$	$t(2) = -1.64$	$p = 0.12$	G:$M = 39.1, \sigma = 13.7$ W:$M = 67.4, \sigma = 13.7$	$t(18) = -5.11$	$\mathbf{p < 0.001}$
Change-simple	G:$M = 29.6, \sigma = 29.6$ W:$M = 52.8, \sigma = 24.2$	$t(25) = -3.72$	$\mathbf{p < 0.001}$	G:$M = 50.4, \sigma = 40.3$ W:$M = 42.4, \sigma = 14.4$	$t(25) = 0.13$	$p = 0.55$
Change-complex	G:$M = 69.9, \sigma = 19.1$ W: *Insufficient data*	–	–	G:$M = 100.4, \sigma = 34.6$ W:$M = 143.7, \sigma = 18.3$	$t(9) = -3.27$	$\mathbf{p = 0.005}$
Compare-groups	G:$M = 38.6, \sigma = 14.0$ W:$M = 102.6, \sigma = 36.8$	$t(28) = -8.54$	$\mathbf{p < 0.001}$	G:$M = 111.0, \sigma = 35.8$ W:$M = 126.5, \sigma = 60.0$	$t(23) = 0.43$	$p = 0.33$
Conflict-simple	G:$M = 54.7, \sigma = 24.6$ W:$M = 104.1, \sigma = 81.0$	$t(20) = -3.23$	$\mathbf{p = 0.002}$	G:$M = 72.8, \sigma = 41.0$ W:$M = 103.1, \sigma = 64.0$	$t(18) = -1.25$	$p = 0.11$
Conflict-complex	G:$M = 29.4, \sigma = 12.2$ W: *Insufficient data*	–	–	G:$M = 52.4, \sigma = 37.3$ W: *Insufficient data*	–	–
Memogate	G:$M = 19.8, \sigma = 15.3$ W:$M = 66.1, \sigma = 49.5$	$t(32) = -5.41$	$\mathbf{p < 0.001}$	G:$M = 105.1, \sigma = 42.0$ W:$M = 116.5, \sigma = 72.0$	$t(28) = -0.46$	$p = 0.33$
Precedence	G:$M = 42.3, \sigma = 24.6$ W:$M = 117.6, \sigma = 108.2$	*No test*	*No test*	G:$M = 70.6, \sigma = 29.3$ W:$M = 115.4, \sigma = 59.7$	*No test*	*No test*

may be that Windows participants gained more fluency with their interface than Grid participants did during the small-scale tasks, which, due to limitations in our study design, always came before the large-scale tasks. Second, Windows has in place more advanced search functionality, which may show greater gains as policies get larger. The first explanation may be a point in the Grid's favor, because it suggests that policy authors gained fluency with the Grid quite quickly, so there was little gain to be had through experience. The second explanation suggests that more effort should be put into developing more advanced search functionality for the Grid. Accurately interpreting the interface*scale effect requires further investigation.

While we have shown that Grids outperform the Windows interface for accuracy and speed, it is worth considering some of the limitations of our study methodology. We gave participants artificial goals in a fictitious scenario with which they were not already familiar. However, in real policy-authoring scenarios, goals often arise intermittently and relate to the author's own environment, with which they are familiar. Thus, some usability problems that arose in our study, such as searching for files, may not be significant problems in real scenarios, because, for example, authors may already know where all of their own files are stored. On the other hand, real policy authors might encounter usability problems, such as difficulty refining goals, that were not revealed in our study, since we gave participants very specific goals. Another limitation of our study methodology is that it measures accuracy relative to an artificial goal; a better metric for the efficacy of a policy-authoring interface might be whether it allows authors to write policies that match their own true intentions. In scoring our data to determine accuracy, we ignored any extraneous rules authors made on their way to completing a task (e.g., allowing or denying some access to a user not mentioned in the task statement), because such extraneous rules were rare in our study and usually came about when a participant misinterpreted a task statement. Grid participants created extraneous rules in 8 of 271 tasks that were scored as correct, and Windows participants created extraneous rules in 4 of 183 tasks that were scored as correct, so these extraneous rules would have had little effect on our results if we had scored them as incorrect. In reality, however, such extraneous rules might create serious security vulnerabilities.

In spite of its apparent advantages over the Windows interface, the Grid interface does have room for improvement. We observed three main types of errors participants made with the Grid interface. First were off-by-one errors, in which the participant had lined up the mouse in a desired column, then moved it to find the desired row, only to have the mouse slip, unnoticed, from the desired column into an adjacent column. Participants then made the right changes to the wrong principal. The second common type of error, which was responsible for the lower-than-expected Grid accuracy scores in the conflict-simple tasks, occurred in searching for resources. Some resources in our teaching assistant scenario had similar names, for example, a folder called "Assignment 1" and a file called "assignment1.pdf." Some participants, when searching for the keyword "assignment," came across the "assignment1.pdf" file first and assumed they had found the correct resource, although the task required changes to the folder. The third common type of error resulted from the

90-degree angle of the text in the tree at the top of the Grid interface. Participants reported that this rotated text was difficult to read. This error occurred, for example, in our small-scale compare-groups task, in which participants had to see if there was any overlap between 2007's TAs and 2006's students. Some participants correctly read who the TAs were, and correctly opened the students group to view its members, but simply overlooked the name of the overlapping member, even though it was visible.

These three errors suggest improvements that could be made to the Grid interface. Several participants explicitly asked for the ability to lock the Grid interface's crosshairs on a given row or column, so that they could then find a desired item in the other dimension without the mouse slipping off the locked row or column. Also helpful in this regard might be some variant on a focus + context visualization, in which a desired region of the grid could be made much larger than its surroundings. A focus region would make for a larger mouse target, and hence for more accurate mouse positioning. The second error could be addressed with improved search features. Although our Grid interface has a search bar and supports search, its display of multiple search hits is inconspicuous, and was overlooked by many users. If they searched for the keyword "assignment," there would be two hits, but they may have noticed only one. In contrast, Windows has a much better developed search feature for displaying multiple search hits—it shows all search hits in a list with summary details of each hit. Only one Windows participant made the error of mistaking the "assignment1.pdf" file for the "Assignment 1" folder compared with five Grid participants who made this error. Finally, the vertical text in the Grid interface is too difficult to read. A number of solutions may address this problem: changing the angle to 45 degrees would probably help somewhat, the letters could be rotated so that words could be read top-to-bottom, or there could be an option to rotate the whole tree to be horizontal when it is not needed to label the grid columns.

CONCLUSION

The results of our user study demonstrate that our Expandable Grid interface for authoring file permissions policies is superior to the Windows XP native file-permissions interface. The results also strongly suggest that the Grid's superior performance was due to its features that were designed to overcome deficiencies in the list-of-rules model. We have thus provided strong evidence that, from a usability perspective, the Expandable Grid approach to policy-authoring interface design is preferable to the list-of-rules approach.

ACKNOWLEDGMENTS

We wish to thank Julie Downs, Joel Greenhouse, Bob Kraut, Howard Seltman, Steve Sheng, and an anonymous reviewer for help with our statistical analyses; Carolyn Brodie, Clare-Marie Karat, John Karat, and Peter Malkin, who provided SPARCLE, the project that inspired the Expandable Grids concept; and Jason Hong, Cynthia Kuo, and anonymous reviewers for comments on early drafts of this paper. This work was supported in part by National Science Foundation Cyber Trust Grants CNS-0433540 and CNS-0627513, and U.S. Army Research Office contract no. DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems") to Carnegie Mellon University's CyLab.

REFERENCES

1. X. Cao and L. Iverson. Intentional access management: Making access control usable for end-users. In *Proc. of the Second Symposium on Usable Privacy and Security (SOUPS 2006)*, pages 20–31, 2006.
2. N. S. Good and A. Krekelberg. Usability and privacy: a study of Kazaa P2P file-sharing. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems(CHI 2003)*, pages 137–144, New York, NY, April 2003.
3. J. Karat, C.-M. Karat, C. Brodie, and J. Feng. Privacy in information technology: Designing to enable privacy policy management in organizations. *International Journal of Human-Computer Studies*, 63(1-2):153–174, July 2005.
4. B. W. Lampson. Protection. *Operating Systems Review*, 8(1):18–24, January 1974. Reprint of the original from *Proceedings of the Fifth Princeton Symposium on Information Sciences and Systems* (Princeton University, March, 1971), 437-443.
5. R. A. Maxion and R. W. Reeder. Improving user-interface dependability through mitigation of human error. *International Journal of Human-Computer Studies*, 63(1-2):25–50, July 2005.
6. M. C. Mont, R. Thyne, and P. Bramhall. Privacy enforcement with HP Select Access for regulatory compliance. Technical Report HPL-2005-10, HP Laboratories Bristol, Bristol, UK, January 2005. Available at `http://www.hpl.hp.com/techreports/2005/HPL-2005-10.pdf`. Accessed on January 10, 2008.
7. J. Rode, C. Johansson, P. DiGioia, R. S. Filho, K. Nies, D. H. Nguyen, J. Ren, P. Dourish, and D. Redmiles. Seeing further: Extending visualization as a basis for usable security. In *Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS 2006)*, pages 145–155, 2006.
8. The Open Group Research Institute. Adage system overview. Available at `http://www.memesoft.com/adage/SystemSpec.ps`. Accessed on September 20, 2006.
9. U.S. Senate Sergeant at Arms. Report on the investigation into improper access to the Senate Judiciary Committee's computer system, March 2004. Available at `http://judiciary.senate.gov/testimony.cfm?id=1085&wit_id=2514`. Accessed on January 10, 2008.
10. M. E. Zurko. Adage usability testing results: Formal testing affinity mapping and questionnaire. Available at `http://www.memesoft.com/adage/affinity.ps`. Accessed on September 20, 2006.
11. M. E. Zurko, R. Simon, and T. Sanfilippo. A user-centered, modular authorization service built on an RBAC foundation. In *Proceedings 1999 IEEE Symposium on Security and Privacy*, pages 57–71, Los Alamitos, CA, May 1999.